This is a capstone project written by Anni Zheng, Catherine Hua, and Lynn Miao in group DS76. Anni Zheng was responsible for doing question 1 to 5 and the extra credit, which are statistical and linear regression related; Catherine Hua was responsible for doing question 6 to 8, which are machine learning and prediction related; and Lynn Miao was responsible for doing question 9 and 10 which was recommender system related. We set our alpha-value to 0.05, which is standard for doing significance tests, and we use 18412460, N number of Anni Zheng, for seeding the random number generator. We then loaded two csv files and transformed some of the characteristic data in spotify52kData.csv, such as loudness of a song, to numeric type and all of the ratings data in starRatings.csv to numeric type (see line 69 to 84 of our python file). Please note that we also left our note comments in performing data imputation and removal in line 87 to 109 of our python file, however, we keep the data cleaning process specific to each question.

1) **D:** We first extracted song duration and popularity data and mapped them as scatterplots to have a basic inspection of whether the data points of song duration and popularity would form an obvious trend. We then chose Mann-Whitney U test to see if there was a statistically significant relationship between these two data by comparing the resulting p-value to the predetermined alpha-value. We also did a linear regression of the two data once the presence of the relationship was confirmed and sketched the regression line to the scatterplot. We used the coefficient of the linear regression line to determine that the relationship between them was negative.
Y: We did a basic visual inspection first to understand if there was an obvious relationship between song duration and popularity, however, our judgment on this step was more of a personal choice and had no influence on our conclusion. We then followed the map from the slides and chose Mann-Whitney U test to give a flexibility that the variances of the two data were not homogenous and the means were not a good tool to measure the two data by the presence of some outliers visualized in the scatter plot. And, since the p-value of the test was smaller than the alpha-value we set, we inspected the linear relationship between the two data using linear regression because it was simple, fast, and would give a direct answer by just looking at the direction where the line tilts. Please note that Anni used ChatGPT for aiding her drawing of the linear regression line to the graph throughout.
F/A: The p-value after doing the Mann-Whitney U test was (or should be almost) 0, which indicated a possibly strong relationship between duration and popularity of a song, and which we may reject the null hypothesis that there was no relationship between the two data. The linear regression line we mapped was tilted down (Figure Q1). The coefficient of the line was around -1.010, which indicated a negative relationship between song duration and popularity data. It is also reasonable to question the validity of the significance and the linear regression line as we may see from the scatterplot that there is a huge gap between the number of available data of short and long songs, so bias might be presence, or the data are not representative (Figure Q1).

2) **D:** We extracted the explicity and popularity data of the songs from the file. We performed a row-wise removal of the N/A data, and we separated the popularity data into groups of explicit and non-explicit songs and plotted them as a scatterplot and distinguished the two groups using two different colors (blue and orange). We then did another Mann-Whitney U test to see if there was a statistically significant relationship between song's explicity and popularity. We inspected the p-value as it rejected the null hypothesis, and we calculated and compared the mean popularity of both groups if the explicity rated songs are being more popular.

Y: Row-wise removal of N/A data was performed instead of imputation to ensure data integrity, and since we wanted to plot data as pairs of [x,y] values, we only used data in which both x and y values were present. Again, the visualization is just for us to have a basic understanding of the characteristics of the two groups of popularity data. We chose Mann-Whitney U test for allowing the variances of the two data to be not homogenous. Since the p-value of the test was smaller than the alpha-value we set, we inspected the mean of the popularity of both groups and compared them to determine if explicitly rated songs are being more popular. Please note that we understood the susceptibility of mean value to outliers, and we allowed the presence of them because both group's popularity data were evenly distributed along the y-axis or popularity scale (Figure Q2).
F/A: The p-value of the Mann-Whitney U test was around 3.0678e-19, which was smaller than the alpha value 0.05, so we reject the null hypothesis that there is no statistically significant relationship between explicity and popularity. We then checked the mean popularity of explicit song, which was around 35.813, and the mean popularity of non-explicit song, which was around 32.791, and stated that the popularity of the explicit song was more popular than that of the non-explicit songs. We noticed that the number of non-explicit songs was many more than that of explicit songs by just looking at Figure Q2 (the coverage of orange data plot was larger), so the risk of getting non-representative non-explicit songs' popularity data was present. And since the difference between 32.791 and 35.813 wasn't large, we might also recommend checking other factors involved.

3) **D:** We extracted the mode and popularity data of the songs from the file. We performed a row-wise removal of the N/A data, and we separated the popularity data into groups of major and minor keys and plotted them as a scatterplot and distinguished the two groups using two different colors (blue and orange). We then did Mann-Whitney U test to see if there was a statistically significant relationship between song's mode and popularity. We inspected the p-value as it rejected the null hypothesis, and we calculated and compared the mean popularity of both groups if the song's mode is related to its popularity.
Y: Again, row-wise removal of N/A data was performed instead of imputation to ensure data integrity, and since we wanted to plot data as pairs of [x,y] values, we only used data in which both x and y values were present. And, again, the visualization is just for us to have a basic understanding of the characteristics of the two groups of popularity data. We chose Mann-Whitney U test for allowing the variances of the two data to be not homogenous. Since the p-value of the test was smaller than the alpha-value we set, we inspected the mean of the popularity of both groups and compared them to determine if major key songs are being more popular. Please note that we understood the susceptibility of mean value to outliers, and we allowed the presence of them because both group's popularity data were evenly distributed along the y-axis or popularity scale (Figure Q3).
F/A: The p-value of the Mann-Whitney U test was around 2.0175e-6, which was smaller than the alpha value 0.05, so we reject the null hypothesis that there is no statistically significant relationship between mode and popularity. We then checked the mean popularity of major key songs, which was around 33.706, and the mean popularity of minor key songs, which was around 32.758, and stated that the popularity of the major key song was more popular than that of the minor key songs. We noticed that the number of major key songs, which is 32,391, is a lot more than the number of minor key songs, which is 19,609, and one may also inspect that from the coverage of orange plots in Figure Q3. That is to say, the risk of getting non-representative minor key songs' popularity data was present. And since the difference between 33.706 and 32.758 wasn't large, we might also recommend checking other factors involved.

4) **D:** We extracted the mentioned 10 features and popularity data of the songs from the file. In a loop, we did an 80%-20% train-test split with random seed with popularity data and each feature data. We built 10 linear regression models by fitting the training data and getting the predictions from the testing data with RMSEs calculated along. We sorted the RMSEs in a table based on ascending RMSE and visualized the scatterplot of instrumentalness and popularity data with their linear regression line, because instrumentalness was the best predictor given the model having the least RMSE.
Y: The 80%-20% train-test split was inspired by the requirement from project 2. For simplicity, we extended the use of this ratio to the final project. We also used random seed to ensure our results were reproducible. We used RMSE for measuring the performances because it was standard in many performance tests, and we were concerned about the error or residual of our prediction to the actual outcome. We sorted all RMSE values in a one table to give a straight-forward ranking of the model prediction performance (Figure Q4.1) and visualized the best linear regression model to allow ourselves to eyeball the trend made by the two data and if the line of the best fit fits the trend (Figure Q4.2).
F/A: We used RMSEs for measuring how well we could predict the popularity. Please see Figure Q4.1 for the RMSEs of the linear regression model built using 10 different predicting features. We noticed that instrumentalness is the feature with the least RMSE of around 21.591, so we stated that instrumentalness predict popularity best. However, inspecting Figure Q4.2, we saw that the tilt of the line was not obvious, and the regression line was closed to the mean, so we would also make a claim that there is no difference in making a prediction using a simple mean than using a linear regression, so we wouldn't think that the linear regression model was good. We also noticed that the difference of RMSEs between the best and the second-best model was small (around 0.167), so neither do we think that the performance of using instrumentalness to predict popularity was outstanding.

5) **D:** We extracted the mentioned 10 features and popularity data of the songs from the file and performed a row-wise removal of the N/A data. We then did an 80%-20% train-test split with random seed using the features and the popularity data. And we built a multiple linear regression model by fitting the training data and getting the prediction from the testing data. We calculated the RMSE of the model prediction and actual outcome from the testing data. After that we built two more models – we built Ridge and Lasso models to prevent overfitting of our input training data. We used hyperparameter tuning when doing the regularization, and we also calculated the RMSEs of the two regularized models. Lastly, we compared the RMSE of all models we had in the table and sorted them based on the ascending RMSE.
Y: Row-wise removal of N/A data was performed instead of imputation and element-wise removal to ensure data integrity in the prediction (we viewed the predictors and predictions as pairs). The 80%-20% train-test split was inspired by the requirement from project 2. For simplicity, we extended the use of this ratio to the final project. We also used random seed to ensure our results were reproducible. We used RMSE for measuring the performance because it was standard in many performance tests, and we were concerned about the error or residual of our prediction to the actual outcome. We chose hyperparameters ranging from 0 to 100 for ridge regression and 0.01to 100 for lasso regression since they covered a large range of alpha-values for allowing the predictions to be tuned. And we sorted every RMSE value we got in one table to give a straight-forward ranking of the model prediction performance (Figure Q5). Please note that Anni used ChatGPT for knowing that GridSearchCV can be used to do regularized regression.
F/A: We used RMSE for measuring how well we could predict the popularity. The RMSE of multiple linear regression was 21.276 and the same for ridge-regularized and lasso-regularized models. In fact, they were the top three or the three models with the least RMSE values among all available models. The

greatest RMSE difference was between that of multiple linear regression model and that of tempo model with a difference of around 0.528. Please inspect more RMSEs in the table to see how the models are improved (Figure Q5). Since the RMSEs of the multiple linear regression model and that of the two regularized models were the same – regularization did not help a lot in improving prediction. The improvement in prediction with more than one predictor was due to the control of many confounders and the model is being more specific in this case.

6) **D:** For this question, we used PCA to extract the principal components. First, we identified the number of genres in the dataset, which was 52. After standardizing the data, we performed a PCA on the selected data and picked the major components using the elbow method along with eigenvalues. Using the PCA-transformed dataset with 4 major components, we computed the silhouette for searching the number of clusters in the dataset.
Y: The PCA was adopted to reduce dimensions and identify clusters based on the transformed dataset. Standardizing the data was often necessary to perform a PCA. The threshold of "eigenvalue>1" corresponded to the component that explained more variance than an individual input. We used the silhouette scores to find the optimal number of clusters within the transformed data.
F/A: Judging from the eigenvalues, there were in total 4 components that had eigenvalues greater than 1. The result can be observed from Figure 6.1. The selected components explained 67.341% of the variance, accounting for most of the dataset. As Figure 6.2 displayed the decrease of the scores when the number of clusters increased, the highest silhouette score was 0.36 when k=3, which indicated 3 clusters based on the search. The result was not close to the designated clusters, for the clusters do not correctly correspond to the 52 genres; as observed from Figure 6.3, the songs were roughly identified into three categories. This reflected using PCA-transformed data might not be the best approach when finding a large number of clusters as there was a loss of information due to the reduction.

7)**D:** For this question, we used a logistic regression for initial predictions. We preprocessed the data by checking null values and standardizing them. Then we split the data into 80/20.To improve the result, we investigated the performance of a more advanced model, Decision Tree.
Y: The logistic regression was executed as a primary strategy for classification. The dataset was split to avoid overfitting. Based on the initial result, we considered another model to improve the performance.
F/A: From Figure 7.2, we can see the accuracy and precision were relatively good but not ideal. The poor performance was reinforced by Figure 7.1, where the AUC score only reached 0.5, as the two lines almost overlapped, indicating valence does not predict whether a song is in major or minor well. As Figure 7.3 shows the AUC score as 0.55, there was a slight increase in the performance when using Decision Tree. Decision Tree is able to do feature selection, understands the given data better than Logistic Regression, and produces a better classification. Another possible approach was to include more features to predict the target, as the single variable may not be representative or informative enough to allow a good classification.

8)**D:** For this question, we built a neural network with hidden layers to perform a multi-classification using the ten features. We set the number of hidden units to 100, batch size to 32, learning rate to 0.001, and lambda_l2 to 1e-3. The detail of the model is shown as follows:
Sequential(
(0): Linear(in_features=10, out_features=100, bias=True)

(1): Linear(in_features=100, out_features=100, bias=True)
(2): Linear(in_features=100, out_features=52, bias=True)
)

Then we transformed the data into tensors to feed into neural networks and trained the model based on the batches. We also recorded the training and validation loss to observe the learning curve. Finally we computed the AUC score using the mode "One - vs.- Rest".

Y: To increase performance, we set the hidden units to 100 with a batch size of 32. The learning rate of 0.001 allowed for gradual and stable learning, as it is a conventional value used in neural networks. The multiple hidden layers also improved the final results. We set "bias=True" to add a bias term to help the model fit better. The lambda value was set to 1e-3 to regularize the model and prevent overfitting. The loss curves were used to observe the training process of the network.

F/A: As Figure 8.1 depicts, the loss decreases as the number of epochs increases and decreases slower around epochs = 50. Given the training loss and validation loss were close, the learning of the model was good. The final AUC score we retrieved for "One - vs.- Rest" was 0.806, representing a good multi-classification result for all 52 genres. Potentially, there might be better hyperparameters to increase our results using GridSearch, but the process would be time-consuming.

9A) **D:** First we calculated the average star rating for the 5k songs in the star dataset, and then we created a new dataframe called "popularity," which takes the popularity values from the first 5000 songs, since only the first 5000 songs are explicitly rated. We then created a simple linear regression model to use popularity to predict star_rating.

Y: We used a linear regression model to see how much of the variance in star_rating could be predicted by popularity.

F/A: The R^2 score of the model is 0.324, which means that 32.4% of the variance in star_rating can be explained by popularity, and the two are positively correlated (see appendix for plot).

9B) **D**: For the average star ratings, we sorted the values in descending order, and chose the top 10, and did the same with the "popularity" dataframe that contains the popularity values from the first 5000 songs.

Y: There are two possible ways to find the top 10 songs based on the popularity based model, depending on your definition of popularity. We can define popularity based on the average star rating of the song, or with the "popularity" measure provided by Spotify, which corresponds to the number of plays. The top 10 songs by ratings and popularity are in the appendix.

F/A: Of the top 10 by popularity and ratings, only two songs overlap - 2003 and 3003. We will move forward with defining popularity based on listens, so Spotify's popularity measure, since explicit feedback is rare (in this case only exists for 5000/52000 songs), and there are concerns with selection bias in explicit feedback.

10) To create a recommender system, we used collaborative filtering with a latent factor model, using gradient descent to update the user and item matrices. First we took the star dataframe, and randomly initialized user and item matrices. We set the hyperparameters and latent factors, with k=2 latent factors, a learning rate of 0.01, and with 380 epochs. We also set our batch size to 100, in order to optimize the code's speed. We converted all NaNs to zeros to prepare for calculations for gradient descent. Then we use a series of for loops: in the innermost for loop, we predict the actual rating and the predicted rating

using the user and item factor vectors, and calculate the prediction error. We then calculate the gradients for the user and item factor vectors in order to minimize the cost function, here defined as the sum of the squared differences between actual and predicted ratings. We also subtract a regularization term to prevent overfitting, and then we update the user and item matrices by scaling the learning rate and moving the direction of the gradients. We do this for all users for each item, and we do this 380 times, the number of epochs. Ideally, we could have also tuned the hyperparameters and increased the number of epochs, but due to runtime, we could only conduct this iteration.

To find the final set of predicted ratings, we take the dot product of the user and item matrix. We then find the top 10 ratings per user for songs they haven't rated. To do so, first we melted the predicted and star ratings so that each row is a user, song, and rating. Then we drop all NAs from the star ratings to filter down to songs that the user has rated. We do a left join to keep only predictions where the user has not rated the song. We then use groupby and apply a function to return the top 10 predicted songs for each user that are left in the prediction dataframe.

To see how our recommended songs compared to the "greatest hits" in question 9B, we took all the top 10 songs we recommended to each user, and found the top 10 most recommended songs across all 10k users, and then compared that list with the "greatest hits," and found that only 2/10 songs overlapped - songs 2003 and 3003. While we would expect that the most popular songs would likely appear in our recommendations, given the strength of the popularity baseline, we hope that the other songs are a result of our improvement on the baseline. Our top 10 recommended songs are in the appendix.

To evaluate how good our model is overall, we relied on precision to evaluate our recommendations - essentially, how often is a song that we recommend in the top 10 for each user is "relevant" to that user? First, we identified our top 10 recommendations for each user, regardless of whether or not they had rated it previously. We used a lambda function across our predicted ratings dataframe to find the 10 highest rated songs for each user, and then melted the data such that each row was a user, song, and their rating. Then, we created a flag to mark if a song was "relevant" to a user based on their ratings - if they rated it a 3 or a 4, the two highest ratings, we flagged it as relevant with a 1, otherwise 0. Our next step is to merge this dataframe of our predicted ratings with the actual ratings with an inner join, so that our merged dataframe only contains the top 10 recommendations for each user, and if the user actually rated the song. We then found the percentage of top 10 recommendations that are "relevant" to our users, by summing the number of 1s (our relevant flag), over the total number of top 10 recommendations we made that the users had rated.

We found that 54% of our top 10 recommendations were "relevant" and rated a 3 or 4 by the users. We also examined the actual distribution of 3 or 4 star ratings in the star data, and found that only 39% of the ratings were 3 or 4. Thus, our recommender system likely surfaced a tailored, relevant mixtape for each of our 10k users.

Extra Credit) We are interested in knowing the presence of a relationship between the number of capital letters of album name and song popularity.

D: We first extracted the number of capital letters of all album names and the corresponding popularity data. Please note that we disregarded all non-Latin letter album names by setting the minimum number of

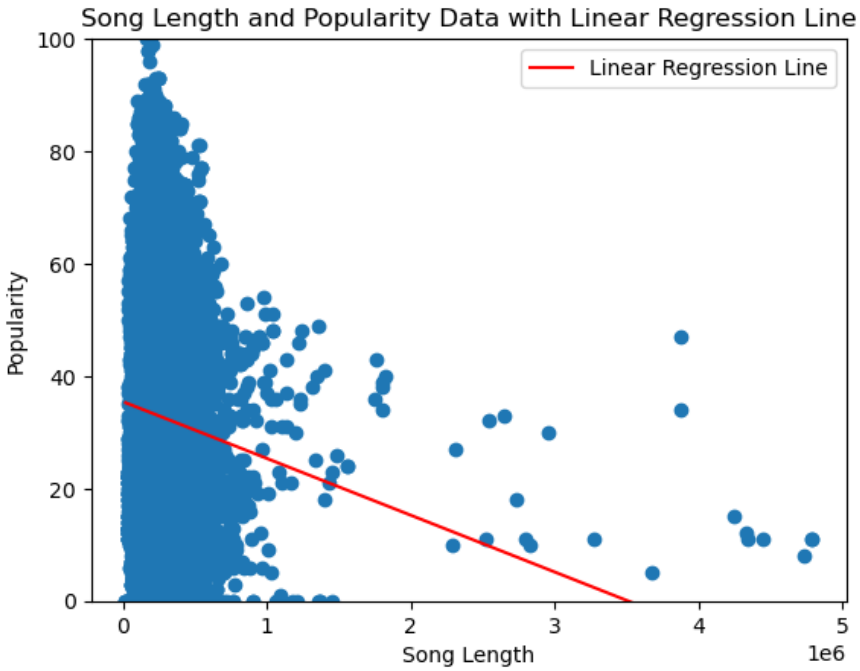
capital letters we accounted for to be 1. We were also taking the risk in losing some Latin letter albums with no capital letter. We then separated the popularity data into groups based on the numbers of capital letters of the corresponding albums. And we chose to use ANOVA to perform the significance test of these groups and compare the resulting p-value to the alpha-value. Unfortunately, the p-value was so high that we were unable to reject the null hypothesis. We also visualized the popularity and number of capital letters data in a figure to eyeball how unrelatable the two features were (Figure Extra Credit).

Y: We did a basic visual inspection first to understand if there was an obvious relationship between song duration and popularity, however, our judgment on this step was more of a personal choice and had no influence on our conclusion, but we did find this graph to be a little bit similar that of Figure Q1 probably because most of the music creators chose to make their song's duration and capital letter under a fixed range, so that they were not too outstanding. We used an ANOVA test because we were having more than two groups of data (there were 35 groups with the number of capital letters ranging from 1 to 38). For simplicity, we disregarded any non-Latin album by setting the minimum capital letter to 1 for each data we accounted for. The resulting p-value we got is close to 1, so it was mostly unlikely that there was a relationship between the number of capital letters in an album and its popularity. Please note that Anni used outside sources to know a short-cut to find the uppercase letter in a string and a short-cut in unpacking groups in a dictionary.

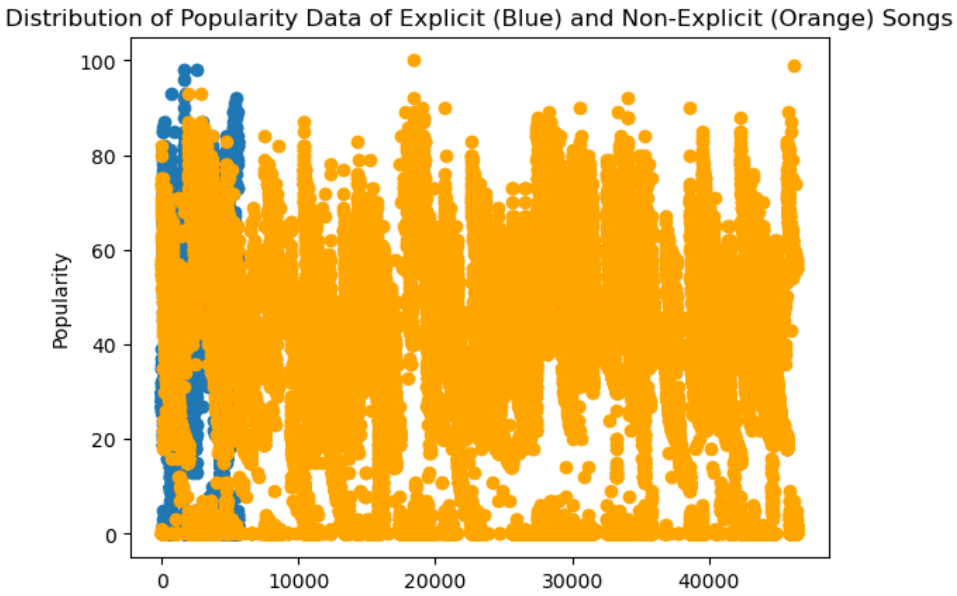
F/A: The p-value after doing the ANOVA test was close to 1 indicating that we could not use the number of capital letters in an album to predict its popularity. However, we did recognize the risky take we had in disregarding any album with no capital letter, because there was a possibility that albums with no capital letters were (un)popular. It was also naïve to inspect only albums with Latin letters.

Appendix

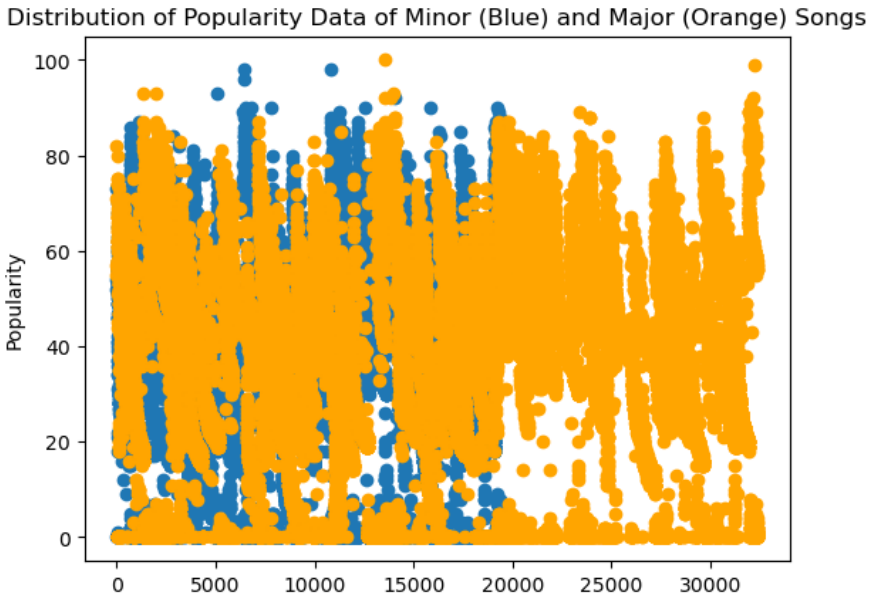
Q1 Song Length and Population Data with Linear Regression Line



Q2 Distribution of Popularity Data of Explicit and Non-Explicit Songs



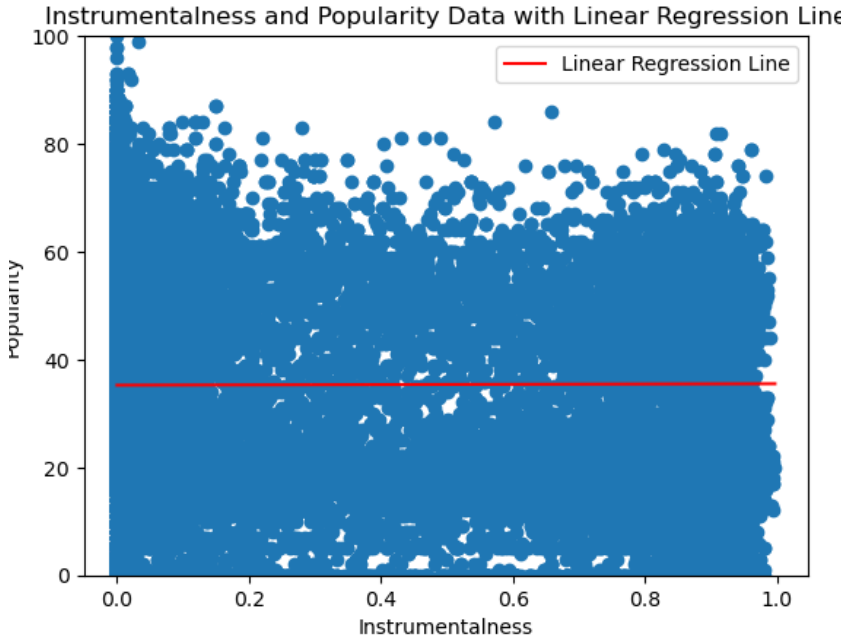
Q3 Distribution of Popularity Data of Minor (Blue) and Major (Orange) Songs



Q4.1 Performance (RMSE) of Features in Predicting Popularity

Feature	RMSE
instrumentalness	21.590923
loudness	21.758559
duration	21.767838
energy	21.770325
valence	21.781651
speechiness	21.783831
liveness	21.788992
danceability	21.793547
acousticness	21.796968
tempo	21.803471

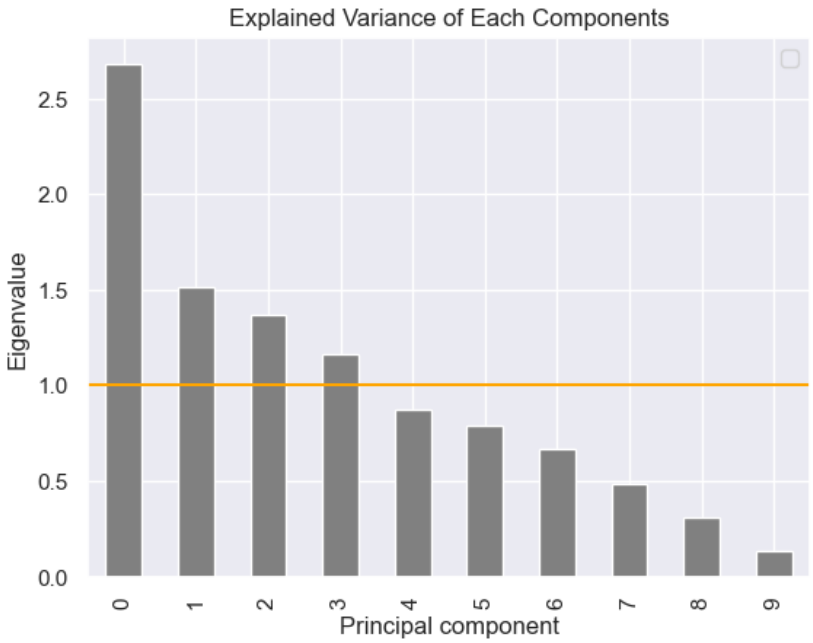
Q4.2 Instrumentalness and Popularity Data with Linear Regression Line



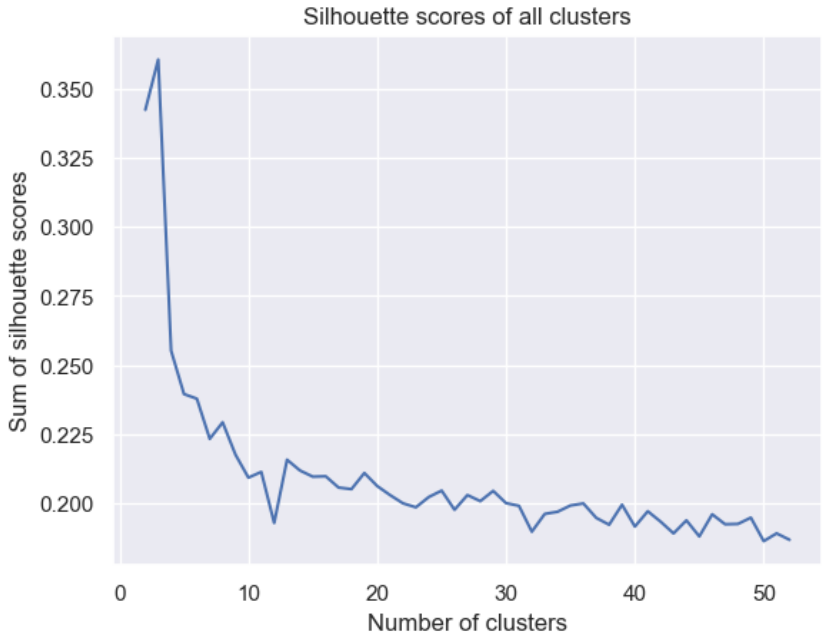
Q5 Performance (RMSE) of Predictors (Single and Multiple) in Predicting Popularity

Predictors	RMSE
Multiple Predictors	21.275923
Multiple Predictors with Ridge Regression	21.275932
Multiple Predictors with Lasso Regression	21.275932
instrumentalness	21.590923
loudness	21.758559
duration	21.767838
energy	21.770325
valence	21.781651
speechiness	21.783831
liveness	21.788992
danceability	21.793547
acousticness	21.796968
tempo	21.803471

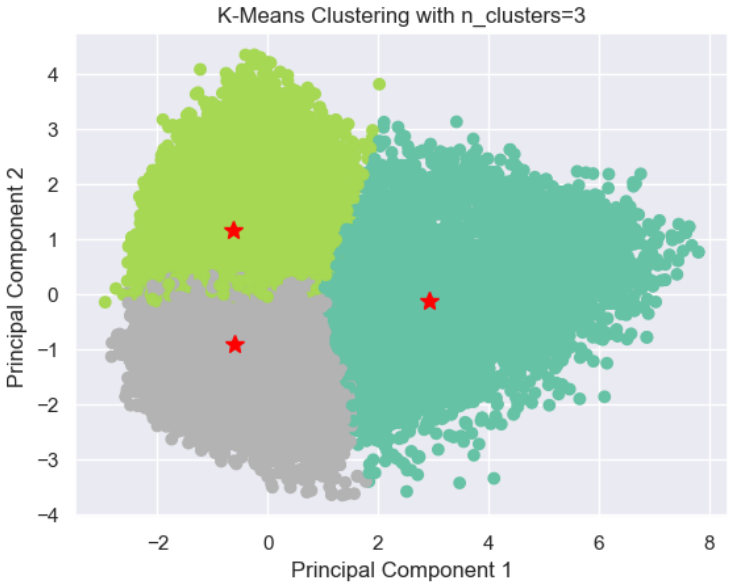
Q6.1 Explained Variance of Each Component



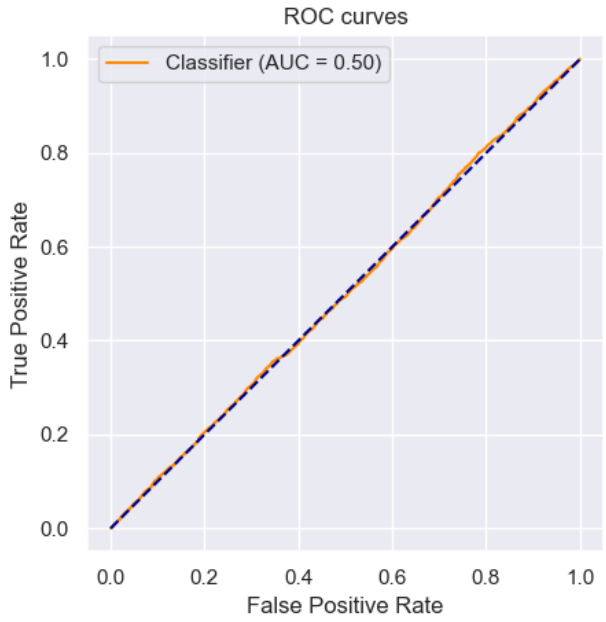
Q6.2 Changes in the Silhouette Scores



Q6.3 Kmeans Clusters Using the Transformed Data



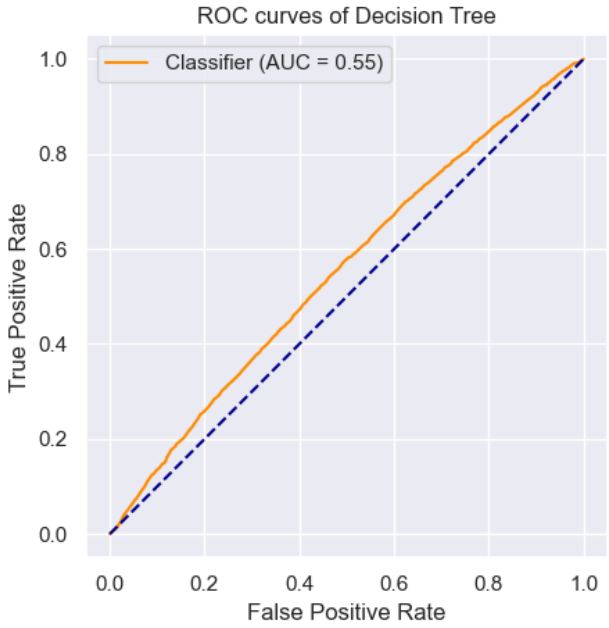
Q7.1 ROC Curve of Logistic Regression



Q7.2 Classification Report of Logistic Regression

```
Accuracy = 61.6%
Confusion matrix:
[[   0 3996]
 [   0 6404]]
Precision = 61.6%
Recall = 100.0%
```

Q7.3 ROC Curve of Decision Tree

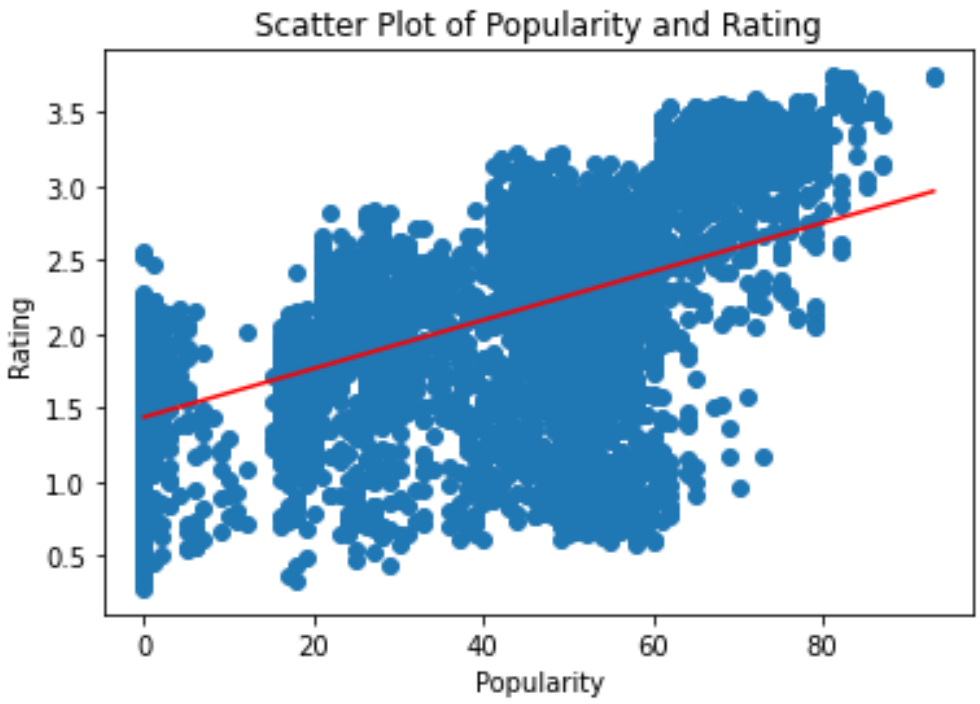


Q7.4 Classification Report of Decision Tree

	precision	recall	f1-score	support
0	0.45	0.16	0.23	3996
1	0.63	0.88	0.73	6404
accuracy			0.60	10400
macro avg	0.54	0.52	0.48	10400
weighted avg	0.56	0.60	0.54	10400

Q8.1 Training and Validation Loss of Neural Network

Q9A Scatterplot



Q9B: Top 10

Top 10 Songs by Rating

Song	Avg Rating
3877	3.75
3003	3.74895
2260	3.744554
2562	3.743202
3216	3.741969
2105	3.737475
2003	3.729651
2011	3.729124
3464	3.727829
3253	3.727451

Top 10 Songs by Popularity

Song	Popularity
2003	93
3003	93
3300	87
2000	87
3000	87
2106	86
3004	86
2002	86
3257	86
3002	86

Q10

Top 10 Recommended Songs from our Recommender

Song	Counts
3003	7491
3877	7311
2260	7136
3007	6749
2003	6653
3464	6059
2011	5456
2954	4926
2257	4767
2105	4516

Extra Credit